BPA Software CR Specs

Software CR

The Software CR is another component of the Binary Provisioning Agent(BPA). The Software CR will install the required software, drivers and perform software updates on the hosts that make up the kubernetes cluster after the BPA controller has successfully installed kubernetes on the host. The required software and versions are specified in the software CR under the master and worker sub sections respectively. Software in the master section will be installed on all master nodes in the cluster and software in the worker section will be installed on all worker nodes in the cluster. If the software version is not specified, the latest version in that repository will be installed.

When the Software CR is created and the Kubernetes cluster has successfully been installed, the BPA controller looks up the matching software custom resource (**cluster name in the Software CR must match cluster name in the provisioning CR**), gets the software list and then installs the software (and /or updates) that are defined in the hosts via SSH. It gets the IP address for each host from the IP address configmap created by the BPA controller when it was installing the cluster

Software CRD

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: software.bpa.akraino.org
spec:
  group: bpa.akraino.org
  names:
    kind: software
    listKind: softwarerList
    plural: software
    singular: software
    shortNames:
    - sw
  scope: Namespaced
  subresources:
    status: {}
  validation:
    openAPIV3Schema:
      properties:
        apiVersion:
          description:
          type: string
        kind:
          description:
          type: string
        metadata:
          type: object
        spec:
          type: object
        status:
          type: object
  version: v1alpha1
  versions:
  - name: v1alpha1
    served: true
    storage: true
```

Software CR Object Definition(*_types.go)

The software_types.go file is the API for the software Custom resource

```
package v1alpha1

import (
        metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

// EDIT THIS FILE!  THIS IS SCAFFOLDING FOR YOU TO OWN!
// NOTE: json tags are required.  Any new fields you add must have json tags for the fields to be serialized.

// SoftwareSpec defines the desired state of Software
// +k8s:openapi-gen=true
type SoftwareSpec struct {
        // INSERT ADDITIONAL SPEC FIELDS - desired state of cluster
        // Important: Run "operator-sdk generate k8s" to regenerate code after modifying this file
        // Add custom validation using kubebuilder tags: https://book-v1.book.kubebuilder.io/beyond_basics
/generating_crd.html
        //MasterSoftware map[string][]interface{} `json:"masterSoftware,omitempty"`
        MasterSoftware []interface{} `json:"masterSoftware,omitempty"`
        WorkerSoftware []interface{} `json:"workerSoftware,omitempty"`
        //WorkerSoftware map[string][]interface{} `json:"workerSoftware,omitempty"`
}

// SoftwareStatus defines the observed state of Software
// +k8s:openapi-gen=true
type SoftwareStatus struct {
        // INSERT ADDITIONAL STATUS FIELD - define observed state of cluster
        // Important: Run "operator-sdk generate k8s" to regenerate code after modifying this file
        // Add custom validation using kubebuilder tags: https://book-v1.book.kubebuilder.io/beyond_basics
/generating_crd.html
}

// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

// Software is the Schema for the softwares API
// +k8s:openapi-gen=true
// +kubebuilder:subresource:status
type Software struct {
        metav1.TypeMeta   `json:",inline"`
        metav1.ObjectMeta `json:"metadata,omitempty"`

        Spec   SoftwareSpec   `json:"spec,omitempty"`
        Status SoftwareStatus `json:"status,omitempty"`
}

// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

// SoftwareList contains a list of Software
type SoftwareList struct {
        metav1.TypeMeta `json:",inline"`
        metav1.ListMeta `json:"metadata,omitempty"`
        Items           []Software `json:"items"`
}

func init() {
        SchemeBuilder.Register(&Software{}, &SoftwareList{})
}
```

MasterSoftware : This corresponds to the masterSoftware section in the sample CR file below. It has a list of interfaces containing the various software, if the version is specified, the item is a map corresponding to software name and version, if it is not the item is a string that has the software name

WorkerSoftware: It is also as described in the masterSoftware

Sample Software CR YAML files

```
apiVersion: bpa.akraino.org/v1alpha1
kind: Software
metadata:
  labels:
    cluster: cluster-xyz
    owner: c1
  name: example-software
spec:
  masterSoftware:
    - curl
    - htop
    - jq:
        version: 1.5+dfsg-1ubuntu0.1
    - maven:
        version: 3.3.9-3
  workerSoftware:
    - curl
    - htop
    - tmux
    - jq
```

Cluster IP Address Configmap

In order for the BPA controller to install the software specified in the software CR via SSH, it would need to have saved the IP addresses of the hosts in that cluster. The BPA controller creates a configmap that stores the hosts in a cluster and their corresponding IP address. Below is an example of the configmap the BPA operator creates;

Cluster IP address configmap

```
apiVersion: v1
data:
  MASTER_master-1: 192.168.30.20
  WORKER_worker-1: 192.168.30.65
kind: ConfigMap
metadata:
  creationTimestamp: "2019-09-24T21:03:32Z"
  labels:
    cluster: cluster-abc
  name: cluster-abc-configmap
  namespace: default
  resourceVersion: "8519202"
  selfLink: /api/v1/namespaces/default/configmaps/cluster-abc-configmap
  uid: c426f1fc-df0e-11e9-a107-00219ba0c77a
```